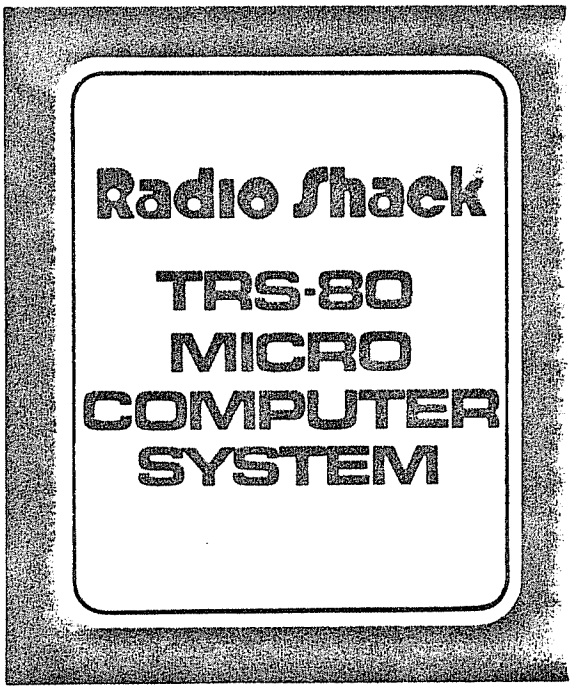
LINK-80 Reference Manual



For Use with the TRS-80
Disk Operating System (TRSDOS)

LINK-80 Linking Loader
Reference Manual

Contents

SECTION 2

LINK-80 Linking Loader

The LINK-80 Linking Loader takes the relocatable object files generated by the FORTRAN compiler and MACRO-80 assembler and loads them into memory in a form that can be executed. In addition, LINK-80 automatically searches the system library (FORLIB) and loads the library routines needed to satisfy any undefined global references (i.e., calls generated by the compiled program to subroutines in the system library).

LINK-80 provides the user with several loading options. Programs may be loaded at user-specified locations, and program areas and data areas may be separated in memory. A memory image of the executable file produced by LINK-80 can be written to disk. The default extension for the name of the executable file is /CMD.

1.1 Running LINK-80

When you give TRSDOS the command

```
L80
```

(diskette #2 must be in the disk drive), you are running the LINK-80 linking loader. When the loader is ready to accept commands, it prompts the user with an asterisk. The loader will exit back to TRSDOS after a command containing an E or G switch (see Section 2.1.1), or after a <break> is done at command level.

Command lines are also supported by LINK-80.

1.1.1 LINK-80 Commands

A command to LINK-80 consists of a string of filenames and/or switches. The command format is:

```
[filename1][-switch1][,filename2][-switch2]...
```

All filenames must be in TRSDOS filename format.

After LINK-80 receives the command, it will load or search (see the S switch) the specified files. Then it will list all the symbols that remained undefined, with each followed by an asterisk.

*=TEST	Examine file TEST/CRF and generate a cross reference listing file TEST/LST.
*T=TEST	Examine file TEST/CRF and generate a cross reference listing file T/LST.

Cross reference listing files differ from ordinary listing files in that:

1. Each source statement is numbered with a cross reference number.

2. At the end of the listing, variable names appear in alphabetic order along with the numbers of the lines on which they are referenced or defined. Line numbers on which the symbol is defined are flagged with '#'.

Example:

```
*MAIN

 DATA    5200    5300

 SUBR1*      (SUBR1 is undefined)

 DATA    5200    5300

*SUBR1
*-G          (Starts Execution - see below)
```

Typically, to execute a FORTRAN program and subroutines, the user types the list of filenames followed by -G (begin execution). Before execution begins, LINK-80 will always search the system library (FORLIB/REL) to satisfy any unresolved external references. If you wish to first search libraries of your own, append the filenames that are followed by -S to the end of the loader command string.

1.1.2 LINK-80 Switches

A number of switches may be given in the LINK-80 command string to specify actions affecting the loading process. Each switch must be preceded by a dash (-). These switches are:

Switch	Action
R	Reset. Put loader back in its initial state. Use -R if you loaded the wrong file by mistake and want to restart. -R takes effect as soon as it is encountered in a command string.
E or E:Name	Exit LINK-80 and return to the Operating System. The system library will be searched on the current dsk to satisfy any existing undefined globals. The optional form E:Name (where Name is a global symbol previously defined in one of the modules) uses Name for the start address of the program. Use -E to load a program and exit back to the monitor.
G or G:Name	Start execution of the program as soon as the current command line has been interpreted. The system

library will be searched on the current disk to satisfy any existing undefined globals. Before execution actually begins, LINK-80 prints two numbers and a BEGIN EXECUTION message. The two numbers are the start address and the address of the next available byte. The optional form G:Name (where Name is a global symbol previously defined in one of the modules) uses Name for the start address of the program.

N

If a <filename>-N is specified, the program will be saved on disk under the selected name (with a default extension of CMD) when a -E or -G is done.

P and D

-P and -D allow the origin(s) to be set for the next program loaded. -P and -D take effect when seen (not deferred), and they have no effect on programs already loaded. The form is -P:<address> or -D:<address>, where <address> is the desired origin in the current typeout radix. (Default radix is hexadecimal. -O sets radix to octal; -H to hex.) LINK-80 does a default -P:<link origin> (i.e., 5200).

If no -D is given, data areas are loaded before program areas for each module. If a -D is given, all Data and Common areas are loaded starting at the data origin and the program area at the program origin. Example:

```
*-P:200,FOO
Data    200    300
*-R
*-P:200 -D:400,FOO
Data    400    480
Program 200    280
```

U

List the origin and end of the program and data area and all undefined globals as soon as the current command line has been interpreted. The program informa-

tion is only printed if a -D has been done. Otherwise, the program is stored in the data area.

M — List the origin and end of the program and data area, all defined globals and their values, and all undefined globals followed by an asterisk. The program information is only printed if a -D has been done. Otherwise, the program is stored in the data area.

S — Search the filename immediately preceding the -S in the command string to satisfy any undefined globals.

Examples:

```
*-M                          List all globals

*MYPROG,SUBROT,MYLIB-S
                    Load MYPROG.REL and SUBROT.REL and
                     then search MYLIB.REL to satisfy
                     any remaining undefined globals.

*-G                          Begin execution of main program
```

1.2 Sample Link

```
DOS READY
L80
*EXAMPL,EXMPL1-G
DATA    5200    52AC
[5200   52AC]
[BEGIN EXECUTION]

    1792                    14336
   14336                   -16383
  -16383                       14
      14                      112
     112                      896
DOS READY
```

1.3 Format of LINK Compatible Object Files

NOTE

Section 2.3 is reference material for users who wish to know the load format of LINK-80 relocatable object files. Most users will want to skip this section, as it does not contain material necessary to the operation of the package.

LINK-compatible object files consist of a bit stream. Individual fields within the bit stream are not aligned on byte boundaries, except as noted below. Use of a bit stream for relocatable object files keeps the size of object files to a minimum, thereby decreasing the number of disk reads/writes.

There are two basic types of load items: Absolute and Relocatable. The first bit of an item indicates one of these two types. If the first bit is a 0, the following 8 bits are loaded as an absolute byte. If the first bit is a 1, the next 2 bits are used to indicate one of four types of relocatable items:

00 Special LINK item (see below).

01 Program Relative. Load the following 16 bits after adding the current Program base.

10 Data Relative. Load the following 16 bits after adding the current Data base.

11 Common Relative. Load the following 16 bits after adding the current Common base.

Special LINK items consist of the bit stream 100 followed by:

a four-bit control field

an optional A field consisting of a two-bit address type that is the same as the two-bit field above except 00 specifies absolute address

an optional B field consisting

of 3 bits that give a symbol
length and up to 8 bits for
each character of the symbol

A general representation of a special LINK item is:

```
1 00 xxxx  yy nn    zzz + characters of symbol name
           --------  ------------------------------
             A field             B field
```

xxxx	Four-bit control field (0-15 below)
yy	Two-bit address type field
nn	Sixteen-bit value
zzz	Three-bit symbol length field

The following special types have a B-field only:

0	Entry symbol (name for search)
1	Select COMMON block
2	Program name
3	Request library search
4	Reserved for future expansion

The following special LINK items have both an A field and a B field:

5	Define COMMON size
6	Chain external (A is head of address chain, B is name of external symbol)
7	Define entry point (A is address, B is name)
8	Reserved for future expansion

The following special LINK items have an A field only:

9	External + offset. The A value will be added to the two bytes starting at the current location counter immediately before execution.
10	Define size of Data area (A is size)
11	Set loading location counter to A
12	Chain address. A is head of chain, replace all entries in chain with current location counter. The last entry in the chain has an address field of absolute zero.
13	Define program size (A is size)
14	End program (forces to byte boundary)

The following special Link item has neither an A nor a B field:

15	End file

1.4 LINK-80 Error Messages

LINK-80 has the following error messages:

?No Start Address	A -G switch was issued, but no main program had been loaded.
?Loading Error	The last file given for input was not a properly formatted LINK-80 object file.
?Out of Memory	Not enough memory to load program.
?Command Error	Unrecognizable LINK-80 command.
?<file> Not Found	<file>, as given in the command string, did not exist.

%2nd COMMON Larger /XXXXXX/

The first definition of COMMON block /XXXXXX/ was not the largest definition. Reorder module loading sequence or change COMMON block definitions.

%Mult. Def. Global YYYYYY

More than one definition for the global (internal) symbol YYYYYY was encountered during the loading process.

%Overlaying [Program / Data] Area [,Start = xxxx / ,Public = <symbol name>(xxxx) / ,External = <symbol name>(xxxx)]

A -D or -P will cause already loaded data to be destroyed.

?Intersecting [Program / Data] Area

The program and data area intersect and an address or external chain entry is in this intersection. The final value cannot be converted to a current value since it is in the area intersection.

?Start Symbol - <name> - Undefined
After a -E: or -G: is given, the symbol specified was not defined.

Origin [Above / Below] Loader Memory, Move Anyway (Y or N)?
After a -E or -G was given, either the data or program area has an origin or top which lies outside loader memory (i.e., loader origin to top of memory). If a Y <cr> is given, LINK-80 will move the area and continue. If anything else is given, LINK-80 will exit. In either case, if a -N was given, the image will already have been saved.

?Can't Save Object File
A disk error occurred when the file was being saved.

1.5 Program Break Information

LINK-80 stores the address of the first free location in a global symbol called $MEMRY if that symbol has been defined by a program loaded. $MEMRY is set to the top of the data area +1.

NOTE

If -D is given and the data origin is less than the program area, the user must be sure there is enough room to keep the program from being destroyed. This is particularly true with the disk driver for FORTRAN-80 which uses $MEMRY to allocate disk buffers and FCB's.